Development Guidelines

A handbook created by developers for developers to help engineering teams align their software practices and share know-how.

Abstract

This is a high-level document containing a collection of best practices, commonalities between projects and values proven to be practical. Team's should follow these guidelines when implementing their software.

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119.

These guidelines are valid for all application development in Fintraffic Cloud environments.

Table of Contents

1 Codebase

1.1 Version Control

- MUST use version control
- MUST use Fintraffic GitHub
- MUST use trunk based development or Gitflow
- MUST have documented version control flow

1.1.1 Branching

- SHOULD protect production branch

1.1.1.1 Mobile development

- SHOULD use fast-forward merges only from feature branch to main branch
- SHOULD implement bug fixes to feature branch and cherry picked them to main and potential release branch
- RECOMMENDED to squash feature branches before merging to main branch
- MUST preserve release tags forever

1.2 Peer Review

- SHOULD have a process for peer review
- SHOULD have another developer to approve code changes before executed in production

1.3 Coding Standards

- MUST agree on a coding standard inside a team
- RECOMMENDED use automatic code formatting
- SHOULD use automatic code style checking (linting)
- SHOULD sign code and artifacts
- SHOULD have source control protection
- SHOULD perform smoke tests for own coding
- SHOULD use code quality tools

1.4 Architecture

- MUST have only needed components in production (resources, interfaces, dependencies)
- MUST follow common API Guidelines (when available)
- MUST make your technology choices visible (in TechRadar when available)
- MUST use architectural decision records when doing technical decision-making
- MUST have documented inventory of used components and dependencies

2 Release Management

- SHOULD release to production from main (trunk)
- MUST have identifiable releases
- RECOMMENDED to release smaller changes often over larger merges
- MUST have (at least) following stages in pipeline (in recommended order): test, release (, run)
- SHOULD have (at least) following stages in pipeline (in recommended order): install, test, scan, build, deploy, verify, release
- SHOULD have a defined security quality gate for production releases

2.1 Mobile Development

- MUST use semantic versioning for releases (tags)

3 Environments

3.1 Data

3.1.1 Mobile Development

- SHOULD preserve all release artefacts forever

3.1.2 Test data

- SHOULD use production data in test environments
- MUST protect test environment data like production environment data
- MUST anonymize personal data in test environments

3.2 Design

- SHOULD name AWS profiles after account-aliases
- MUST have production separated from testing environments
- MUST follow the Principle of Least Privilege
- MUST have separation between all environments (e.g. dev/-test-prod)
- MUST conduct threat modeling in the design phase

4 Architecture

- MUST document all integrated 3rd party provided services used by the application
- MUST document selected development management tools and purpose of tools

4.1 Infrastructure

- SHOULD use semantic versioned Container images for building releases
- SHOULD use aliases when versioning Lambdas
- MUST have centralised logging
- SHOULD collect logs from all deployed environments
- MUST collect logs from production environment
- MUST use tags on cloud resources
- MUST have logging on operating system level, application level and network level
- MUST collect access logs
- MUST collect user action logs for admin-level users
- SHOULD have all virtual machines managed by CSP's instance management service (AWS Systems Manager, Azure Automanage, …)
- MUST have all virtual machines for production managed by CSP's instance management service (AWS Systems Manager, Azure Automanage, …)
- SHOULD use cloud managed services whenever possible
- MUST have periodical OS updates for all services not managed by cloud
- MUST document manually managed virtual machines' maintenance and security processes
- MUST encrypt data at rest in cloud via KMS
- MUST rotate encryption keys every 365 days (that are used for data at rest) in cloud
- MUST have billing alerts in cloud
- MUST monitor costs in cloud
- SHOULD have infrastructure as code
- MUST have repeatable infrastructure
- MUST use encryption in transit for all public endpoints

4.2 Security

- SHOULD run automated vulnerability checks for code
- MUST use a dependency checker such as GitHub Dependabot
- SHOULD use hardened software components
- SHOULD use hardened hardware components
- MUST include malware checks as a part of any file upload features
- MUST validate security controls on the server side (not only client side)
- SHOULD run automated static code analysis for code quality
- MUST restrict unwanted access to all environments from the open internet
- MUST follow best practices of AWS and CIS
- MUST push security related production logs to SIEM (when available)
- MUST follow information security requirements in the contract

4.3 Compliance

4.3.1 GDPR

- MUST ensure that company policy regarding cookie classification and consents are followed (if applicable)
- MUST document the application architecture, personal data flows and needed security measures to ensure compliance with business, privacy and information security requirements
- MUST document all personal data used in testing and development, backups and log data

- MUST have required monitoring functionalities concerning usage of personal data based on data classification
- MUST ensure that only necessary personal data will be collected and/or processed
- MUST implement functionality for defining retention times for all personal data
- MUST be able to erase (or anonymize) all (or partial) personal data on expiration or when requested by data subject

4.3.2 Classified Data

- MUST classify processed data according to Fintraffic policy
- MUST follow data policy according to data classification

5 Operations

5.1 Monitoring

- MUST have monitoring defined and implemented
- MUST have alarms defined and implemented
- SHOULD provide uptime metric(s) for a service
- SHOULD use FTCLOUD monitoring template
- SHOULD monitor security metrics
- SHOULD monitor cost metrics
- SHOULD have targeted alerts for service or operations

5.2 Continuity

- MUST have continuity plan
- MUST have disaster recovery plan
- MUST review continuity plan & disaster recovery plan regularly
- MUST test continuity & disaster recovery plan in at least a tabletop exercise
- MUST conduct a Lessons Learned after a security incident

5.3 Backups

- MUST have daily backups
- MUST test backup recovery at least annually
- MUST have backups of the data from production
- MUST store backups separate from production data
- MUST encrypt backups

6 Guidance

- MUST have documented process how to handle security notifications

6.1 Documentation

- MUST use README.md as a central information document inside code repository
- SHOULD document all exceptions with reasoning from Development Guidelines in project's README.md

6.2 Security

- SHOULD have secure coding training for developers
- SHOULD have a channel for developers to request help on security issues
- MUST have time allocated for developers to conduct security fixes

7 Quality Assurance

- SHOULD have documented test strategy
- MUST have strategy for test automation and automatic tests designed based on it

7.1 Automation

- SHOULD develop, use and share common solutions (at least) internally for test automation
- RECOMMENDED to consider existing solutions before looking for a new one
- RECOMMENDED to run tests for all code changes as part of CI/CD pipeline